UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/651,425	08/30/2000	Christopher Songer	003048.P008	2566

23363 7590 06/15/2005

CHRISTIE, PARKER & HALE, LLP
PO BOX 7068
PASADENA, CA 91109-7068

EXAMINER	
VU, TUAN A	
ART UNIT	**PAPER NUMBER**
2193	

DATE MAILED: 06/15/2005

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	09/651,425	SONGER ET AL.	
	Examiner	Art Unit	
	Tuan A. Vu	2193	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *3* MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *12 April 2005*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-44* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-44* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

1. This action is responsive to the Applicant's response filed 4/12/2005.

As indicated in Applicant's response, claims 1, 22, 43-44 have been amended. Claims 1-44 are pending in the office action.

Claim Rejections - 35 USC § 103

2. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

3. Claims 1-17, 19-38, and 41-44 are rejected under 35 U.S.C. 103(a) as being unpatentable over, in view of Tseng al., USPN: 6,009,256 (hereinafter Tseng), in view of Schlansker et al., USPN 6,408,428 (hereinafter Schlansker) and further in view of Trimberger, USPN: 5,752,035 (hereinafter Trimberger) .

As per claim 1, Tseng discloses a method of creating run time executable code, the execution using a processing element array, comprising:

partitioning a processing element (PE) array into a plurality of hardware accelerators (e.g. col. 1, lines 45-59; FPGA *chip*, col. 10, lines 2-22; col. 12, lines 45-55; Fig. 30-33 – Note: array of chips on to be reconfigured into circuits helping the execution of the kernel code is equivalent to array of PEs being partitioned in plurality of hardware accelerators)

decomposing a program source code (e.g. col. 58, line 62 to col. 59, line 50; Fig. 4, 26, 28; *Verilog, VHDL, parsing process, compiler 210* - col. 15, line 62 to col. 16, 38 - Note: language that is programmed from human readable format then parsed and compiled into

machine readable constructs are source program code, e.g. Verilog and VHDL) into a plurality of kernel sections (e.g. *code component* -col. 13, lines 27-36; *HDL code description, component*-- col. 11, lines 47- 64; *RTL level ... signals* – col. 60, lines 53-62; Fig. 29 – Note: HDL code component analysis is equivalent to decomposing into kernel sections; steps *301, 302, 304, 310,* Fig. 4)

(i) mapping said plurality of kernel sections into a plurality of hardware dependent model entities representation language – or hardware dependent executable code (e.g. *hardware model, gate level* – col. 11, lines 47- 64; *RTL* - col. 11, line 50 to col. 12, line 6; *hardware execution models* - col. 13, lines 36-40 - Note: register level code representation reads on hardware dependent executable code because binary code ultimately generated from RTL and hardware model is executable - see step 310 – Fig. 4); and

generating a mapping between combinations of said hardware accelerators and said hardware dependent executable code entities configured to support run time execution (e.g. steps *306, 307, 309* - Fig. 4 ; Fig. 6; col. 19, lines 14-21; col. 21, lines 36-43) of the plurality of kernel sections (e.g. Fig. 25-30), the combinations of accelerators including code variants which perform a function whose inputs and outputs are identical (e.g. Fig. 17, 18a-b; *LATCH 1009, 1010, 1011, 1012* – Fig. 23 – Note: code related to implementing a latching function of a flip-flop reads on performing yielding of an output exactly as it has been at the input).

But Tseng does not explicitly disclose that the mapping in (i) is a matrix mapping of kernel sections into a plurality of hardware dependent executable code. However, Tseng discloses a programming language derived from the hardware dependent model entities, e.g. gate level, for support of execution or simulation/debug on said plurality of hardware accelerators

(e.g. Fig. 25-30; col. 11, line 50 to col. 12, line 6 – Note: RTL level signal used in program to effect signaling of a FPGA and support emulation/simulation/debug of hardware circuit is equivalent to mapped kernel sections code for executing hardware accelerators in FPGA) and partitioning via analysis from a software kernel so as to map during compilation of the kernel into execution by a hardware accelerator among the element array (e.g. col. 11, line 47 to col. 12, line 15) with help of graphical tabular means to enhance gate or registers mapping (*table* – col. 31, lines 36-44; Fig. 7, Fig. 16) in the optimizing process (cols 21-26). Hence, the mapping – like using a table or a mapping spreadsheet or a matrix -- of hardware accelerators to said hardware dependent code to support runtime of the kernel sections by the processing element array is suggested. It would have obvious for one of ordinary skill in the art at the time the invention was made to implement such mapping thus mentioned with the use of a table or a matrix as suggested because such mapping means is used in all mapping endeavor for its benefits in graphical clarity allowing visual understanding.

But Tseng does not disclose identifying a plurality of functions in a program source code that are anticipated to consume a substantial execution time; nor disclose that such identified functions are the plurality of kernel sections. Tseng discloses removing of redundant logic and use of equation for optimizing the anticipated time consuming connectivity paths and minimizing cost functions along critical paths (col. 21, line 44 to col. 22, line 45); hence has suggested optimizing of code by emphasizing on most resource taking paths. Analogous to the concept of mapping software into hardware implemented capability by Tseng, Schlansker, in a method to find an optimum design for match hardware parameters and performance requirements to a particular type of processors, i.e. to translate into processor instruction code the corresponding

parameterization specifics representing a hardware dependent requirements analogous to the mapping of kernels to hardware RTL language just like Tseng (see Tseng - Fig. 4 and related text) recognized otherwise as software/hardware partitioning) , discloses mapping of instruction code sets via intermediate code against processor/hardware parameters -- or kernel sections as claimed (e.g. *parametization, opsets, opgroups* - col. 9, line 24 to col. 10, line 32; *AIR* - col. 19, line 23 to col. 21, line 4; Fig. 12-15) and further discloses organizing instruction code into set and groups for evaluation against operational statistics (e.g. Fig. 16-20 – Note: collection of statistics and metrics for performance evaluation is equivalent to profiling). It would have been obvious for one of ordinary skill in the art at the time the invention was made to map the hardware dependent code to the kernel sections by Tseng so that such mapping is based on test-bench processes; and also on statistical profiling by Schlansker; because with the support of such statistical metrics, the evaluation of a execution capability/resource of particular hardware dependent processing element or processor can be obtained (see Schlansker, col. 3, line 36 to col. 4, line 12). But Schlansker does not particularly specify that the metrics gathering is to anticipate a program code to consume a substantial execution time. Trimberger, in a method for compiling and executing programs using re-programmable logic accelerator sets analogous to the FPGA accelerators used in Tseng's method (combined with Schlansker) for simulation, discloses the use of profiling to detect the most frequently executed code and replace it with programmable instructions unit, or accelerator set (e.g. col. 6, lines 31-56). It would have been obvious for one of ordinary skill in the art at the time the invention was made to enhance the mapping of such code into executable instruction groups (or kernels) as taught by Tseng with the mapping based on resource usage statistics as taught by Schlansker and with Trimberger's

accounting for most frequently executed code, because this would help create a corresponding set of executable that best suits the hardware specification of the accelerators without extraneous use of resources, averting thereby hardware architecture/code operation non-compliance as intended by Schlansker (e.g. col. 3, line 36 to col. 4, line 12) in light of code frequency of execution by Trimberger.

As per claims 2 and 3, the combination Tseng/Schlansker further discloses partitioning into digital signal processors (re claim 2) and into bins (re claim 3) (Tseng: *EAB, DSP* -- col. 51, lines 27-34 ; *cluster* – col. 22, line 46-57; Schlansker: instruction groups – Fig. 16)

As per claim 4, Tseng further discloses mapping includes mapping into multiple hardware contexts (e.g. col. 22, line 12-45; Fig. 28 – Note: grouping by gate netlist/Register Translation, or chip cluster logic/common clock is equivalent to mapping for multiple hardware context).

As per claims 5 and 6, Tseng further discloses mapping (re claim 5) a first set of invariants (e.g. *FPGA component lib 361, logic function 359* – Fig. 6) and that said first set of variants are produced (re claim 6) based on resource usage (e.g. RTL *358*, Fig. 6; *combinational component 303*, Fig. 4 – Note: register allocation is equivalent to resource usage).

As per claim 7, Tseng further discloses mapping a second set of variants of said designs configured to support multiple hardware configurations of one of a plurality of bins (e.g. circuit *design components, gate netlist* – col. 22, lines 1-23).

As per claim 8, Tseng further discloses mapping is performed by a place and route (e.g. steps *352, 353, 354, 355* – Fig. 6).

As per claim 9, Tseng further discloses the decomposition step is performed manually (*user* -- e.g. col. 17, lines 13-17).

As per claim 10, Tseng does not specify that the decomposition step is performed by a software profiler; but discloses deriving of gate combinational elements based on type analysis (e.g. col. 18, lines 22-41; step 302, Fig. 4) for network array execution path differentiation; and profiling functions to optimize the cost for placement of processing elements in the array (e.g. col. 23, line 58 to col. 23, line 21; Fig. 6). As mentioned in claim 1, Schlansker discloses organizing instruction code into set and groups for evaluation against operational statistics (e.g. Fig. 16-20 – Note: collection of statistics and metrics for performance evaluation is equivalent to profiling) while Trimberger discloses the use of profiling to detect the most frequently executed code and replace it with programmable instructions unit, or accelerator set (e.g. col. 6, lines 31-56). The motivation to provide a profiler to Tseng's decomposing into kernel would have been obvious because of the same reasons as set forth in claim 1 in regard to anticipating code execution that is most consuming in time.

As per claim 11, Tseng discloses that the decomposing step includes executing code from program source code and monitoring timing of said execution (e.g. col. 20, line 4 to col. 21, line 24; Fig. 6; *monitor($time...), test-bench component (monitor) 906* – Fig. 26,28; *Eval Timer 1004*, Fig. 23).

As per claims 12 and 13, Tseng discloses utilizing set of test data (e.g col. 13, lines 41-47; *evaluate test-bench components*, steps *333, 337*-Fig. 5; Figs. 29-30) in the execution of the simulation; but does not specify using test data (re claim 12) in the decomposing step nor does Tseng specify that (re claim 13) said monitoring includes determining functions that consume a

significant portion of execution timing. But the limitation of using a profile analysis to determine code portions occupying significant execution time prior to optimization has been addressed in claim 10 above in view of Schlansker and Trimberger's teachings. In view of Tseng's teaching to partition software simulation and hardware simulation (Figs. 29-30) using iterative test-bench components testing as mentioned above, it would have been obvious for one of ordinary skill in the art at the time the invention was made to modify the decomposition step the use of test data as taught by Tseng and combine it with the profiling technique as taught by Trimberger combined with Schlansker for detecting the code portion susceptible for further optimizing, or for accelerating in Tseng's method , because this would enhance further the incremental the code optimization or simulation cycle as taught by the combination Tseng (combined with Schlansker/Trimberger's teaching).

As per claim 14, Tseng discloses a decomposition step with identifying functions by identifying regular structures (e.g. Fig. 26-33 – Note: code implementing a write/read function with clock, wire, edge and pins read on identifying regular structures).

As per claim 15, Tseng discloses identifying kernel sections by identifying functions with a limited number of inputs and outputs by way of basic blocks (e.g. col. 44, lines 55-65; Figs. 17,18) of the hardware model, which are register component types fetched from the component type analysis, i.e. decomposition step into basic blocks (e.g. col. 18, lines 20-41— Note: basic blocks being functions chunk of statements are those whose inputs and outputs connection are in very limited number).

As per claim 16, Tseng further discloses identifying functions with a limited number of branches (e.g. col. 18, lines 20-41; col. 44, lines 55-65 - Note: a skill in the art would view basic blocks as those having very limited number of branching)

As per claim 17, Tseng discloses decomposing by identifying overhead sections (e.g. software model 215, Fig. 3; col. 16, lines 28-32; step 302 – Fig. 4).

As per claim 19, Tseng further discloses that mapping includes creating context dependent configurations (e.g. col. 22, line 12-45; components *901, 903, 904*-Fig. 28 – Note: configuring by gate netlist/Register Translation language, or chip cluster logic/common clock context is equivalent to mapping for multiple context dependent configuration; code components are equivalent to code context configurations).

As per claims 20 and 21, Tseng does not explicitly teach that the matrix used in the mapping is sparely (re claim 20) or fully (re claim 21) populated; but discloses the connectivity matrix and mapping of FPGA circuits to hardware models (e.g. Fig. 7, 16; col. 25, line 49 to col. 26, line 15), the component type used for defining partial or generalized connectivity in the array of processing elements (e.g. col. 18,lines 20-41). Combining Tseng's teaching with Schlansker/Trimberger mapping a design logic to configuration resources as mentioned in claim 1, the limitation on such matrix being populated as claimed herein would have been obvious because of the same rationale mentioned in claim 1; and also because Tseng's component type analysis as mentioned above would imply sparsely or fully populating of the matrix as mentioned in claim 1.

As per claim 22, this claim is a system claim corresponding to claim 1 above and includes most of the limitations therein using Tseng's disclosure, namely:

a processing element array or plurality of hardware accelerators partitioned (e.g. col. 1, lines 45-59; FPGA *chip*, col. 10, lines 2-22; col. 12, lines 45-55);

plurality of kernel sections (e.g. *HDL language, code component* -col. 13, lines 27-36; *HDL code description, component*-- col. 11, lines 47- 64)

created from a program source code (e.g. col. 58, line 62 to col. 59, line 50; Fig. 4, 26, 28; *Verilog, VHDL, parsing process, compiler 210* - col. 15, line 62 to col. 16, 38) for execution on said plurality of hardware accelerators (e.g. Fig. 25-30);

plurality of hardware dependent executable code (e.g. *hardware model, RTL, gate level,* – col. 11, lines 47- 64; *hardware execution models* - col. 13, lines 36-40);

(i) mapping of combinations of said hardware accelerators and kernel designs for run time execution (e.g. steps *306, 307, 309* - Fig. 4 ; Fig. 6; col. 19, lines 14-21; col. 21, lines 36-43), the combinations of accelerators including code variants which perform a function whose inputs and outputs are identical (e.g. Fig. 17, 18a-b; *LATCH 1009, 1010, 1011, 1012* – Fig. 23 – Note: code related to implementing a latching function of a flip-flop reads on performing yielding of an output exactly as it has been at the input);

hence is rejected using the corresponding rejection as applied in claim 1 above, using Tseng's teachings.

However, like in claim 1, Tseng does not explicitly teach the mapping in (i) is done using a matrix for mapping of kernel sections into a plurality of hardware dependent executable code for execution on a plurality of hardware accelerators and that said hardware dependent executable code is to support runtime execution of the kernel sections. But this limitation has been addressed in claim 1.

Nor does Tseng disclose identifying a plurality of functions in a program source code that are anticipated to consume a substantial execution time; nor disclose that such identified functions are the plurality of kernel sections.. This limitation has been addressed in claim 1 using Tseng's teachings, in view of Schlansker and Trimberger as set forth above.

As per claims 23-30, these are system claims corresponding to claims 2-9, respectively; hence, are rejected using the corresponding rejections set forth therein, respectively.

As per claims 31-38, these claims are system claims corresponding to claims 10-17, respectively, hence, are rejected herein using the corresponding rejections set forth therein, respectively.

As per claims 41-42, these claims are similar to claims 20-21 above, respectively; hence are rejected herein using the same grounds set forth therein.

As per claim 43, this claim is a computer-readable medium version of claim 1, above hence includes all the step limitations therein and is rejected herein using the same corresponding rejections set forth therein.

As per claim 44, this claim is a system version of claim 1, above hence includes all the step limitations therein and is rejected herein using the same corresponding rejections set forth therein

4. Claims 18, 39, and 40 are rejected under 35 U.S.C. 103(a) as being unpatentable over Tseng et al., USPN: 6,009,256, in view of Schlansker et al., USPN 6,408,428, in view of Trimberger, USPN: 5,752,035; as applied to claims 1 and 22 above, and further in view of Mirsky et al., USPN: 6,457,116 (hereinafter Mirsky).

As per claim 18, Tseng (combined with Schlansker/Trimberger) does not disclose that mapping includes creating microcode. Mirsky, in the method for configuring array of processing elements in response to state data submitted analogous to the simulation and testing of test-bench components applied to the circuit of processing elements disclosed by Tseng, discloses the use of microcode (e.g. col. 10, lines 21-32). In view of the use resource-constrained processing elements such as chips and DSP as suggested by Tseng (col. 51, lines 27-34) for partitioning, it would have been obvious for one of ordinary skill in the art at the time the invention was made to implement microcode to the processing element such as taught by Mirsky to Tseng's (combined with Schlansker/ Trimberger) mapping of hardware configuration to the design circuit for carrying out the simulation process as taught by Tseng. One of ordinary skill in the art would be motivated to do so because this would alleviate memory storage in small device used as PEs in the array of Tseng's system (combined with Schlansker/ Trimberger), and further improve resource usage and time-efficiency.

As per claim 39, this claim corresponds to claim 18 above, hence, is rejected herein using claim 18 rejection as set forth above.

As per claim 40, Tseng discloses code including context dependent configurations (e.g. *register component 901, clock component 903, Test-bench component 904* – Fig. 28 – Note: all component-related code with configuration for variables are equivalent to context dependency configurations); but Tseng fails to disclose microcode in place of code. Such limitation has been addressed in claim 18 above and is rejected herein with the same ground of rejection set forth therein.

Response to Arguments

5. Applicant's arguments filed 4/12/2005 have been fully considered but they either moot in view of new grounds of rejection or not persuasive. Following are the most representative points raised from the arguments and corresponding response by Examiner.

(A) Applicants have submitted that Tseng alone or in combination does not teach or suggest "a matrix describing different combinations ... inputs and outputs are identical." (Appl. Rmrks, pg. 11, top para). The rejection has pointed out how the matrix for mapping combination of hardware accelerators to hardware dependent executable code has been considered obvious in light of the similar form of teaching in Tseng, RTL mapping and gate mapping using matrix or tabular forms. The code variants are construed as just entities of the mapping matrix that take into account a functionality of a combination of accelerators such that the function yields an output data being exactly the same as its input data, i.e. a latch which upon a trigger enabler release the held input at the output port. The argument appear to allege that the reference does not teach or suggest a claimed invention without pointing out specifics as to how the language of the claims patentably distinguishes over the prior art; hence Applicants have not fulfilled the responsibility incumbent to the Applicants as set forth in CFR §1.111, which to point out the specific distinctions believed to have rendered the claims patentable over any applied references.

(B) Applicants have submitted that Trimberger's 'most frequently executed code' is not the same as code 'anticipated to consume a substantial execution time' (Appl. Rmrks, pg. 11, middle para). The purpose of optimizing is to make appropriate code restructuring in order to allow code when executed uses the least resource as possibly as can be done. For example, in an iteration wherein repeated portions of code use the same access to an unchanging piece of data, code should be changed to obviate redundant steps lest execution resources are misused.

Usually, code that consumes a big chunk of runtime is (i) code for large data computations or (ii) for complex loops with highly intricate executions/iterations; and such code is anticipated to consume execution time and resource and might necessitate timely implementing of optimization profiling or analysis. So for one skill in the art when faced with either scenario (i) or (ii), true should be the recognition that runtime resources would most likely be taxed. It is hence reasonable to say that if analysis and techniques of the likes of profiling or code instrumentation or hooks enable a determination that frequently executed code amounts to but a very small portion of runtime resource, one skill in the art would not consider such code as *anticipated to consume* runtime resources simply because optimization is well-known to consider or anticipate scenario in which only code execution that take substantial time or resources, not the opposite scenario. Hence, the argument that code frequently executed might not take as much runtime resource as code 'anticipated to consume substantial execution time' does not seem to take into consideration the very simple reason behind why code needs to be optimized. Trimberger teaches how to address substantial code execution resource taking into consideration most frequently code in the useful context of optimization analogous to Schlansker's method, i.e. not for the mere sake of statistically determining which code is most frequently executed like disregarding the effect of such execution frequency upon runtime limitation.

(C) Applicants have submitted that the use of 3 references amount to awkward combinative steps (Appl. Rmrks, pg. 11, last para). The current rejection has combined Tseng' teaching about bench-test and Schlansker's statistical metrics on resource usage as well as Trimberger's use of most frequently code execution; and has set forth the rationale as to why one skill in the art would want to combine Tseng's netlist approach and optimizing path cost evaluation (see cols.

21-26) with respect to optimizing the functions when routing the task and using a path evaluation cost equation. In light of the amended claim, the rejection has established grounds as to why such attempt of optimizing the functions would have been obvious in light of the statistical profiling shown by Schlansker and the most frequently executed code by Trimberger. In response to applicant's argument that the examiner has combined an excessive number of references, reliance on a large number of references in a rejection does not, without more, weigh against the obviousness of the claimed invention. See *In re Gorman*, 933 F.2d 982, 18 USPQ2d 1885 (Fed. Cir. 1991). The motivation to combine has been established; and Applicants should show specifics as to why such combination would degenerate in negative purposes, yield adverse effects.

(D) Applicants have submitted that Tseng's RTL signal analysis being considered equivalent to 'matrix mapping of kernel sections ... '(Appl. Rmrks, pg. 11, bottom, pg. 12, top para) is but one more awkward step added to the combinative steps. The claim does not provide specifics as to what exactly *kernel sections* or *hardware dependent executable code* amount to. The rejection as put forth stems for the interpretation of the language of the claim using broad and reasonable interpretation. Hardware dependent code has been analogized to implementation of code constructs evolved from the register and signal analysis and data dependency using the concepts from the kernel sections model; and kernel sections have been analogized with gate, registers components being modeled via in part the RTL representation. The claim from lack of specificity has not lead to one skill in the art to an interpretation more favorable to Applicants' liking, i.e. an interpretation that otherwise would render the claimed limitations possibly more patentably distinct over the references being applied. But alleging that the rejection is composed

of awkward combination of steps does not help the claimed invention to be patentably distinct in view of the requirements of CFR §1.111.

Information Disclosure Statement

6. The information disclosure statement filed 11/12/2004 again does not seem to comply with 37 CFR 1.98(a)(2), which requires a legible copy of each U.S. and foreign patent; each publication or that portion which caused it to be listed; and all other information or that portion which caused it to be listed. It has been placed in the application file, but the information referred to is not provided with corresponding physical copy as per the current record; therein has not been considered.

Specifically, the non-patent documents (6 of them) listed in the form PTO-1449, pg. 1-2, are not found to come with a legible copy for each. The documents are not considered and are marked with 'NC'. In order to expedite the consideration of such missing documents, Applicants are urged to resubmit these copies if such non-patent documents are deemed of some import in the prosecution of the case.

Conclusion

7; **THIS ACTION IS MADE FINAL.** Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37

CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Tuan A Vu whose telephone number is (272) 272-3735. The examiner can normally be reached on 8AM-4:30PM/Mon-Fri.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Kakali Chaki can be reached on (571)272-3719.

The fax phone number for the organization where this application or proceeding is assigned is (571) 273-3735 (for non-official correspondence – please consult Examiner before using) or 703-872-9306 (for official correspondence) or redirected to customer service at 571-272-3609.

Any inquiry of a general nature or relating to the status of this application should be directed to the TC 2100 Group receptionist: 571-272-2100.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

VAT
June 6, 2005

KAKALI CHAKI
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100